MuPAD's solutions to the ISSAC challenge

Walter Oevel*
Stefan Wehmeier†
Paul Zimmermann‡

15th July 1997

Abstract

The computer algebra system MuPAD solves eight out of the ten problems of the challenge. We present these solutions, as well as partial solutions to the remaining problems; those problems that could not completely be solved are marked by an asterisk.

Problem 1

What is the 4 significant digit approximation to the condition number of the 256 by 256 Hilbert matrix?

For any symmetric matrix one has $cond_2(A) = |\lambda_{max}|/|\lambda_{min}|$ whith the eigenvalues $\lambda_{max/min}$ of maximal/minimal modulus. They can be computed by a vector iteration (von Mises) with A and A^{-1}, respectively.

It is rather easy to define the $n \times n$-Hilbert matrix:

```
>> n:=256: M:=Dom::Matrix(): A:=M(n,n,func(float(1/(i+j-1)),i,j)):
```

A numerical computation of the inverse $B = A^{-1} = (\beta_{i,j})$ would be expensive and numerically unstable. Instead, we use an explicit formula for the inverse of the $n \times n$ Hilbert matrix taken from the literature, see [2, example 3.8] or [3]:

$$\beta_{i,j} = (-1)^{i+j} \frac{c_i\, c_j}{i+j-1}\,, \quad c_i := \frac{(n+i-1)!}{((i-1)!)^2\,(n-i)!}\,.$$

```
>> c:=[float((n+i-1)!)/float( (i-1)! )^2/float( (n-i)! ) $hold(i)=1..n]:
   B:=M(n,n,[[(-1)^(i+j)*op(c,i)*op(c,j)/(i+j-1) $hold(i)=1..n] $hold(j)=1..n]):
```

To find the largest eigenvalues of `A` we start the vector iteration with $x = (1, \ldots, 1)$:

*Universität-Gesamthochschule Paderborn, walter@uni-paderborn.de
†Universität-Gesamthochschule Paderborn, stefanw@math.uni-paderborn.de
‡INRIA Lorraine, Paul.Zimmermann@loria.fr

```
>> eps:=10^(-5):
   x:=M(n,1,[1.0 $n]):
   iter:=0:
   repeat
   Axx:=A*x;
   quotients:=Axx[i]/x[i] $hold(i)=1..n;
   x:=(1/Axx::norm(Axx,2))*Axx;
   iter:=iter+1
   until abs((max(quotients)-min(quotients))/min(quotients))< eps end_repeat:
   Aresult:=[min(quotients), max(quotients), iter]:
   print(Unquoted, "Maximal eigenvalue of A is in the interval [".
   expr2text(Aresult[1]).", ".expr2text(Aresult[2])."]");
   print(Unquoted, "Steps needed: ".expr2text(Aresult[3]));
```

```
Maximal eigenvalue of A is in the interval [2.3037995710, 2.3038152170]
```

```
Steps needed: 16
```

To find the reciprocal of the smallest eigenvalue of `A` (= the largest eigenvalue of `B`), we start the vector iteration with $x = (1, -1, 1, \ldots)$. We note that in computing Bx only terms of equal sign have to be added up due to the alternating signs in B and x, whence the matrix–vector multiplication is numerically stable. This pattern will be preserved in the iteration, whence all further steps are numerically stable, too:

```
>> x:=M(n,1,[(-1.0,1.0) $n/2]):
   iter:=0:
   repeat
   Bx:=B*x;
   quotients:=Bx[i]/x[i] $hold(i)=1..n;
   x:=(1/Bx::norm(Bx,2))*Bx;
   iter:=iter+1
   until abs((max(quotients)-min(quotients))/min(quotients))< eps end_repeat:
   Bresult:=[min(quotients), max(quotients), iter]:
   print(Unquoted, "Maximal eigenvalue of B is in the interval [".
   expr2text(Bresult[1]).", ".expr2text(Bresult[2])."]");
   print(Unquoted, "Number of steps needed: ".expr2text(Bresult[3]));
```

```
Maximal eigenvalue of B is in the interval [7.6653673400e388, 7.6653678770\
e388]
```

```
Number of steps needed: 3
```

It just remains to multiply the norms of A and B:

```
>> print(Unquoted, "Therefore the condition number is in the interval [".
   expr2text(Aresult[1]*Bresult[1])."., ".expr2text(Aresult[2]*Bresult[2])."].");

   Therefore the condition number is in the interval [1.7659469990e389, 1.765\
   9591160e389].
```

Problem 2

What is the value of $\int_1^6 x^{x^x}\,dx$ *to 7 significant digits?*

MuPAD provides an adaptive quadrature based on the eighth-order Newton-Cotes formula. This solves the problem very quickly:

```
>> interval:=1,6:
   abserror:=0:
   relerror:=10^(-8):
   intlib::ncquad(func(float(x^(x^x)),x), interval, abserror, relerror);

   [1.102664999e36300, 2.982633828e36286, 417, 0]
```

The Newton-Cotes quadrature is carried out until either the absolute error falls below `abserror` or the relative error falls below `relerror`, whatever happens first. The meaning of the numbers in the list is as follows: the first one is the result, the second one is an estimate of the absolute error. 417 function calls were necessary; the last entry is an error flag and should always be zero.

In the upcoming version, this problem can be solved as follows:

```
>> float(misc::freeze(int)(x^(x^x),x=1..6));

   1.102664999e36300
```

Calling `misc::freeze` turns its argument into a "frozen" or "inert" function, which is necessary here to prevent a symbolic integration of the function.

Problem 3

What is $\sum_{n=1}^{\infty}(n^\pi + n^2 + n^{\sqrt{2}} + 1)^{-1/3}$ *to 14 significant digits?*

The basic idea is: find an expansion

$$s_n = \frac{1}{(n^\pi + n^2 + n^{\sqrt{2}} + 1)^{1/3}} = \frac{c_1}{n^{z_1}} + \frac{c_2}{n^{z_2}} + \cdots, \quad z_1 < z_2 < \ldots$$

such that $\sum_{n=1}^{\infty} s_n = c_1\,\zeta(z_1) + c_2\,\zeta(z_2) + \cdots$ (note $\zeta(z) = \sum_{n=1}^{\infty} n^{-z}$). The expansion is obtained by

$$s_n = \frac{1}{(n^\pi + n^2 + n^{\sqrt{2}} + 1)^{1/3}} = \frac{1}{n^{\pi/3}} \frac{1}{(1+\epsilon_n)^{1/3}}, \quad \epsilon_n := n^{2-\pi} + n^{\sqrt{2}-\pi} + n^{-\pi},$$

where $\epsilon_n \approx 1/n^{\pi-2} \to 0$ as $n \to \infty$. With

$$\frac{1}{(1+\epsilon_n)^{1/3}} = 1 - \frac{\epsilon_n}{3} + \frac{2\,\epsilon_n^2}{9} \mp \cdots + c_p\,\epsilon_n^p + c_{p+1}\,\epsilon_n^{p+1} + O(\epsilon_n^{p+2})$$

one finds

$$s_n = \underbrace{\frac{1}{n^{\pi/3}} \left(1 - \frac{\epsilon_n}{3} \pm \cdots + c_p\,\epsilon_n^p\right)}_{t_n} + (s_n - t_n),$$

where

$$s_n - t_n = \frac{1}{n^{\pi/3}}\, c_{p+1}\,\epsilon_n^{p+1} + \cdots = \frac{c_{p+1}}{n^{\pi/3+(p+1)(\pi-2)}} + \cdots \overset{n\gg 1}{\approx} \frac{c_{p+1}}{n^\alpha}$$

with $\alpha := \pi/3 + (p+1)(\pi-2)$. Summation yields

$$\sum_{n=1}^{\infty} s_n = \underbrace{\sum_{n=1}^{\infty} t_n}_{S_1} + \underbrace{\sum_{n=1}^{N} (s_n - t_n)}_{S_2} + \sum_{n=N+1}^{\infty} (s_n - t_n)$$

with $S_1 = \sum_{n=1}^{\infty} t_n$ given by a sum of ζ-values. We estimate the last sum by the leading order $s_n - t_n \approx c_{p+1} n^{-\alpha}$, whence

$$\left| \sum_{n=N+1}^{\infty} (s_n - t_n) \right| \approx |c_{p+1}| \sum_{n=N+1}^{\infty} \frac{1}{n^\alpha} < \frac{|c_{p+1}|}{\alpha - 1} \frac{1}{N^{\alpha-1}}.$$

Requiring this value to be negligible up to precision *precision*, we determine N from

$$\frac{|c_{p+1}|}{\alpha - 1} \frac{1}{N^{\alpha-1}} \leq \textit{precision}.$$

```
>> DIGITS:=20:
   p:=4:
   precision:=10^(-14):
   sn:=1/n^(PI/3)/(1+eps)^(1/3):
   tn:=expr(taylor(sn,eps,p+1)):
   eps:=n^(-PI)*(n^2+n^sqrt(2)+1):
   optn:=[op(expand(tn))]:
   coefftn:=map(optn,subs,n=1):
   powerstn:=zip(optn,coefftn,func(x/y,x,y)):
   substn:=map(powerstn,subs,n=1+hold(EPS)):
   taylortn:=map(substn,taylor,EPS,2):
   powerstn:=map(taylortn,coeff,1):
   zetastn:=map(powerstn,func(zeta(-x),x)):
```

```
S1:=_plus(op(zip(coefftn,zetastn,_mult))):
hcoefficient:=coeff(taylor(1/(1+EPS)^(1/3),EPS,p+2),p+1):
hpower:=PI/3+(p+1)*(PI-2):
# Now the coefficient of the leading term is determined. #
N:=ceil(exp(1/(hpower-1)*ln(abs(hcoefficient)/precision/(hpower-1)))):
# N term suffice to approximate sum(sn-tn,n=1..infinity): #
S2:=_plus(float(sn-tn)$n=1..N):
DIGITS:=14:
print(Unquoted, "S1=".expr2text(float(S1)),"S2=".expr2text(S2));
print(Unquoted, "Result: ".expr2text(float(S1+S2)));
```

```
S1=29.5808971272370, S2=-8.38765674952610
```

```
Result: 21.1932403777110
```

Problem 4

What is the coefficient of x^{3000} in the expansion of the polynomial

$$(x+1)^{2000}(x^2+x+1)^{1000}(x^4+x^3+x^2+x+1)^{500}$$

to 13 significant digits?

The polynomial — let us call it f here — can also be written as

$$((x+1)^4(x^2+x+1)^2(x^4+x^3+x^2+x+1))^{500}.$$

Therefore we compute

```
>> DIGITS:=13:
   p:=mapcoeffs(poly((x+1)^4*(x^2+x+1)^2*(x^4+x^3+x^2+x+1)),float);

                      2          3          4          5          6          7
    poly(7.0 x + 24.0 x  + 54.0 x  + 90.0 x  + 119.0 x  + 130.0 x  + 119.0 x  +

          8          9          10         11         12
    90.0 x  + 54.0 x  + 24.0 x   + 7.0 x   + 1.0 x   + 1.0, [x])
```

```
>> p15:=p^15: p30:=p15^2: p31:=p30*p: p62:=p31^2: p124:=p62^2: p125:=p124*p:
   p250:=p125^2:
```

Computing p250 by repeated squaring (p250:=p^250) takes a bit more time. The 3000th coefficient of $f = p_{250}^2$ can now be determined as follows:

```
>> print(Unquoted, "The 3000th coefficient is ".expr2text(
   _plus(coeff(p250,1500)^2,2*coeff(p250,i)*coeff(p250,3000-i) $i=0..1499)
   );

                The 3000th coefficient is 3.9739422655800e1426
```

Problem 5

What is the largest zero of the 1000^{th} *Laguerre polynomial to 12 significant digits?*

MuPAD's library offers a procedure `orthpoly::laguerre` that computes the coefficients of Laguerre polynomials; however, it does not offer a method of evaluating a Laguerre polynomial at a given numerical point without computing its coefficients. We therefore have to write a little procedure to solve this problem: `Leval(n,x)` returns the quotient $\frac{L_n(x)}{L'_n(x)}$ which is used for a Newton search.

We use the recursion formula

$$L_n(x) = \frac{2n-1-x}{n} L_{n-1}(x) - \frac{n-1}{n} L_{n-2}(x)\ , \quad L_0(x) = 1\ , \quad L_1(x) = -x+1\ ,$$

$$L'_n(x) = \frac{2n-1-x}{n} L'_{n-1}(x) - \frac{n-1}{n} L'_{n-2}(x) - \frac{1}{n} L_{n-1}(x), \quad L'_0(x) = 0\ , \quad L'_1(x) = -1\ ,$$

the latter following by differentiation of the recursion formula for L_n.

```
>> Leval:=proc(n,x)
   local i,l,l1,tmp,dl,dl1,dtmp;
   begin
   l:=1; l1:=1-x; dl:=0; dl1:=-1;
   for i from 2 to n do
   # compute l1=L_i(x) and dl1=L_i'(x),
     set l=L_{i-1}(x) and dl=L_{i-1}'(x) #
   tmp:=(2*i-1-x)*l1/i - (i-1)*l/i;
   dtmp:=(2*i-1-x)*dl1/i -(i-1)*dl/i - l1/n;
   l:=l1; dl:=dl1; l1:=tmp; dl1:=dtmp;
   end_for;
   l1/dl1
   end_proc:
```

In addition, we write a little program for the Newton iteration using this (numerically stable) iterative scheme for evaluating $L_n(x)$ and $L'_n(x)$:

```
>> Lnewton:=proc(n, start, eps)
   local x, delta;
   begin
   x:=start;
```

```
while abs((delta:=Leval(n,x)))>abs(x)*eps do
x:=x-delta
end_while;
x
end_proc:
```

It is known from theory that the largest zero of the n–th Laguerre polynomial does not exceed $2n+1+\sqrt{4n^2+4n+5/4}$, see [1]. We use this as the starting value for the Newton search, which has to converge monotonically to the largest root of L_n.

```
>> DIGITS:=15: n:=1000:
   print(Unquoted, "The largest zero is ".
   expr2text(Lnewton(n,float(2*n+1+sqrt(4*n^2+4*n+5/4)),10^(-14))));

    The largest zero is 3943.247394845270
```

Problem 6*

Find a a lexicographic grobner basis for the (four) polynomial system: ...

MuPAD's `GROEBNER` package offers the following solution to this problem:

```
>> plist:=[poly(8*w^2+5*w*x-4*w*y+2*w*z+3*w+5*x^2+2*x*y-
   7*x*z-7*x+7*y^2-8*y*z-7*y+7*z^2-8*z+8),
   poly(3*w^2-5*w*x-3*w*y-6*w*z+9*w+4*x^2+2*x*y-
   2*x*z+7*x+9*y^2+6*y*z+5*y+7*z^2+7*z+5),
   poly(-2*w^2+9*w*x+9*w*y-7*w*z-4*w+8*x^2+9*x*y-
   3*x*z+8*x+6*y^2-7*y*z+4*y-6*z^2+8*z+2),
   poly(7*w^2+5*w*x+3*w*y-5*w*z-5*w+2*x^2+9*x*y-
   7*x*z+4*x-4*y^2-5*y*z+6*y-4*z^2-9*z+2)]:
   # convert all elements of the sequence into polynomials #
   groebner::gbasis(plist,LexOrder);
```

Unfortunately, MuPAD does not manage to compute the basis within the amount of time available for a demonstration. However, using the concept of dynamic modules (software integration), MuPAD is able to use the special purpose computer algebra system **Singular** to compute a Groebner basis of the ideal generated by `plist`.

Problem 7

What is $\lim_{x\to 0}\frac{f(g(x))}{g(f(x))}$ *to 9 significant digits?*

The problem can be solved by computing the first summand of the Taylor series of $\frac{f(g(x))}{g(f(x))}$.

To do this, first compute the Taylor series of `f` and `g`; the necessary order be set by increasing the global variable `ORDER`:

```
>> f:=tan(tanh(sin(x))) + tanh(sin(tan(x))) + sin(tan(tanh(x)))
    - tan(sin(tanh(x))) - sin(tanh(tan(x))) - tanh(tan(sin(x)))
    - tan(sinh(tanh(x))) - sinh(tanh(tan(x))) - tanh(tan(sinh(x)))
   + tan(tanh(sinh(x))) + tanh(sinh(tan(x))) + sinh(tan(tanh(x))) :
   g := sinh(tanh(sin(x))) + tanh(sin(sinh(x))) + sin(sinh(tanh(x)))
      - sinh(sin(tanh(x))) - sin(tanh(sinh(x))) - tanh(sinh(sin(x)))
      - tan(sinh(sin(x)))  - sinh(sin(tan(x)))  - sin(tan(sinh(x)))
       + tan(sin(sinh(x))) + sin(sinh(tan(x)))  + sinh(tan(sin(x))) :
   ORDER:=16:
   sf:=series(f,x);

         15
    7769 x         16
    -------- + O(x  )
    3274425


>> sg:=series(g,x);

       11          15
    4 x      16592 x         16
    ----- + --------- + O(x  )
    63       297675
```

Now we use MuPAD's composition operator to calculate the Taylor expansion of $\frac{f \circ g}{g \circ f}$:

```
>> sfg:=(sf @ sg)/(sg @ sf);

    24514478609520577408170967296000000000000000000000/80103448751723203083149 8\

    951509084442801 + O(x)
```

It only remains to convert our symbolic solution into a floating point number:

```
>> DIGITS:=9:
   print(Unquoted,"The result is ".expr2text(float(coeff(sfg,0))));

    The result is 3060352455.0
```

Problem 8*

What is $\int_0^1 x^2 Li_3(\frac{1}{x+1})\,dx$ *where* $Li_n(z)$ *is the polylogarithm function* $(= \sum_{k=1}^{\infty} z^k/k^n)$.

We may interchange sum and integral sign here and obtain

```
>> kthsummand:=int(x^2/((x+1)^k*k^3),x=0..1);
```

```
    1           2           1
--------- - ---------- + ---------- +
   3    4       3    4       3    4
- k  + k    - 2 k  + k    - 3 k  + k

                  /   1          4          4     \
2 exp(- k ln(2)) | ------- + ---------- + --------- |
                 |  3    4       3    4     3    4 |
                 \ k  - k    - 2 k  + k   3 k  - k  /
```

The `exp(ln)` looks ugly, therefore we make MuPAD rewrite the expression:

```
>> kthsummand:=expand(kthsummand);
```

```
    1           2           1           2              8
--------- - ---------- + ---------- + ---------- + ---------- +
   3    4       3    4       3    4   k  3    4    k       3    4
- k  + k    - 2 k  + k    - 3 k  + k   2  (k  - k )  2  (- 2 k  + k )

      8
--------------
 k     3    4
2  (3 k  - k )
```

Obviously we have to test each denominator for singularities:

```
>> operands:=[op(kthsummand)];
```

```
--    1           2           1           2              8
|  ---------, - ----------, ----------, ----------, ----------,
|    3    4       3    4      3    4   k  3    4    k       3    4
-- - k  + k     - 2 k  + k   - 3 k  + k  2  (k  - k )  2  (- 2 k  + k )

      8         --
--------------   |
 k     3    4   |
2  (3 k  - k ) --
```

```
>> map(operands,solve@denom);
```

```
  [{0, 1}, {0, 2}, {0, 3}, {0, 1}, {0, 2}, {0, 3}]
```

This shows that we have to handle the first three summands separately:

```
>> S1:=0:
   for k from 1 to 3 do
       S1:=S1+int(x^2/((x+1)^k*k^3),x=0..1):
   end_for;
   unassign(k):
```

```
  85 ln(2)
  -------- - 145/432
    108
```

It remains to evaluate `sum(kthsummand, k=4..infinity)`, which must be done in several steps:

```
>> S2:=sum(_plus(op(operands,1..3)),k=4..infinity);
```

```
                                  2
  zeta(3)   11 PI
- ------- - ------ + 305/216
     3       108
```

Therefore it remains to compute

```
>> S3:=sum(_plus(op(operands,4..6)),k=4..infinity);
```

which MuPAD does not manage to do.

```
      /      2                  8                  8                          \
      | ------------ + ---------------- + --------------, k = 4..infinity |
  sum |  k   3    4     k        3    4     k     3    4                      |
      \ 2  (k  - k )   2  (- 2 k  + k )   2  (3 k  - k )                     /
```

But at least we manage to produce a numerical solution:

```
>> DIGITS:=15:
   S:=float(_plus(op(operands,4..6))):
   S3:=_plus(subs(S,k=i) $hold(i)=4..50);

    -0.00511541285910104
```

```
>> print(Unquoted, "The numeri al solution is ".expr2text(float(S1+S2+S3))."");

    The numerical solution is 0.21088285950.
```

Problem 9

Find the largest eigenvalue λ to 13 significant digits for the integral equation

$$\int_0^1 \exp(x+y+x^2+xy+y^2+x^2y^2)f(y)dy = \lambda f(x) \ .$$

For brevity, write $g(x,y) := \exp(x+y+x^2+xy+y^2+x^2y^2)$, and $(Tf)(x) := \int_0^1 g(x,y)f(y)dy$.

To evaluate T numerically, we use Gauss-Legendre quadrature $\int_0^1 h(y)\,dy \ \approx \ \sum_{j=1}^n b_j\, h(c_j)$. The abscissae c_j are related to the roots of the nth Legendre polynomial: $P_n(2\,c_j - 1) = 0$:

```
>> n:=15:
   DIGITS:=30:
   Pn:=orthpoly::legendre(n,x);
   roots:=float(misc::freeze(solve)(expr(Pn)));

    {0.848206583410427216200648320774,   -0.724417731360170047416186054614,
     0.724417731360170047416186054614,   -0.570972172608538847537226737254,
    -0.394151347077563369897207370981,    0.937273392400705904307758947 71,
    -0.848206583410427216200648320774,   -0.937273392400705904307758947 71,
     0.987992518020485428489565718586,   -0.987992518020485428489565718587,
    -0.201194093997434522300628303394,    0.570972172608538847537226737254,
    -2.892383906519633860396512212 26e-48, 0.394151347077563369897207370981,
     0.201194093997434522300628303394}


>> c:=sort([op(map(roots,func((x+1)/2,x)))]);

    [0.006003740989757285755217140706 69,    0.031363303799647047846120526 1449,
     0.075896708294786391899675839612 9,     0.137791134319914976291906972693,
     0.214513913695730576231386631373,       0.302924326461218315051396314509,
     0.399402953001282738849685848302, 0.5, 0.600597046998717261150314151697,
     0.697075673538781684948603685 49,       0.785486086304269423768613368627,
     0.862208865680085023708093027307,       0.924103291705213608100324160387,
     0.968636696200352952153879473855,       0.993996259010242714244782859293]
```

Transform `Pn` to `Ppn`, the monic associate of `Pn(2*x)`:

```
>> Ppn:=subs(expr(Pn),x=2*x):

                          3          5          7          9          11         13
      143 x       1001 x     1001 x     1001 x     1001 x     455 x      105 x
    - ---------- + --------- - ------- + ------- - ------- + ------- - -------
      3529785344   155725824   3415040    170752     16704      1392       116

       15
    + x
```

The weights b_j may be computed via the derivative of P_n; the appropriate formula can be found in the literature, see (for example) [1]:

```
>> DPn:=poly(diff(Ppn,x)):
   b:=map(c,func(1/DPn(z-1/2)^2/z/(1-z),z),z);

    [0.015376620998058634177314196788 6, 0.035183023744054062354633708 2253,
     0.053579610233585967505934773342 9, 0.069785338963077157223902397 2555,
     0.083134602908496966776600430240 6, 0.093080500007781105513400280 9332,
     0.099215742663555788228059163221 9, 0.101289120962780636440310099983,
     0.099215742663555788228059163221 9, 0.093080500007781105513400280 9332,
     0.083134602908496966776600430240 6, 0.069785338963077157223902397 2555,
```

```
0.053579610233585967505934773342 9, 0.035183023744054062354633708225 3,
0.015376620998058634177314196788 6]
```

We now approximate Tf by $(\bar{T}f)(x) := \sum_{j=1}^{n} g(x, c_j) b_j f(c_j)$ and consider $(\bar{T}f)(x) = \lambda f(x)$ at the discrete points $x = c_1, \ldots c_n$. We write this as $AF = \lambda F$ with the symmetric matrix

$$A := \left(\sqrt{b_i}\, g(c_i, c_j)\, \sqrt{b_j} \right)_{1 \le i,j \le n}$$

and $F = [\sqrt{b_1}\, f(c_1), \ldots, \sqrt{b_n}\, f(c_n)]$. Like in problem 1, the largest eigenvalue is found by a vector iteration (von Mises):

```
>> M:=Dom::Matrix():
   g:=func(exp(x+y+x^2+x*y+y^2+x^2*y^2),x,y):
   A:=M(n,n,[[sqrt(b[i])*sqrt(b[j])*g(c[i],c[j]) $hold(j)=1..n] $hold(i)=1..n]):
   eps:=10^(-14):
   x:=M(n,1,[1.0 $n]):
   iter:=0:
   repeat
   Axx:=A*x;
   quotients:=Axx[i]/x[i] $hold(i)=1..n;
   x:=(1/Axx::norm(Axx,2))*Axx;
   iter:=iter+1
   until abs((max(quotients)-min(quotients))/min(quotients))< eps end_repeat:
   print(Unquoted, "The eigenvalue is in the interval ".
   expr2text([min(quotients), max(quotients)]));
   print(Unquoted, "Number of iterations: ".expr2text(iter));

    The eigenvalue is in the interval [37.529145560335346749263184795 90, 37.52\
    914556033538156263924861420]

    Number of iterations: 13
```

The precision for the operator norm of the integral operator was checked by starting with lower values of n, which were increased until the norm of the discretized operator was stable within the required precision. It was found that $n = 15$ suffices to approximate the eigenvalue up to 13 digits.

Problem 10

Find the smallest positive number r such that the solution of $\frac{d^2y}{dx^2} = x^3 + y^3 + \frac{dy}{dx}^3$, $y(0) = 0$, $y'(0) = 0$ has a derivative singularity at $x = r$. Calculate r to 13 significant digits. Is $y(r)$ infinite or finite? If $y(r)$ is finite then compute it to 13 significant digits.

A singularity analysis reveals that the solution around blow-up time $x = r$ is of the form

$$y(x) = -\sqrt{2(r-x)} + s(\sqrt{2(r-x)}), \quad y'(x) = \frac{1 - s'(\sqrt{2(r-x)})}{\sqrt{2(r-x)}}$$

with some smooth function s. This motivates the following transformation $(y,p) \to (z,q)$

$$y = z - \sqrt{q}\ ,\ p = \frac{1}{\sqrt{q}} - 1 \quad \Longleftrightarrow \quad z = y + \frac{1}{p+1}\ ,\ q = \frac{1}{(p+1)^2}$$

of the system $y' = p\ ,\quad p' = x^3 + y^3 + p^3$ to a more stable form which can be handled by the Runge-Kutta method:

$$\begin{aligned} z' &= 2 + (1 - x^3 - y^3)\, q - 3\sqrt{q}\ , \\ q' &= -2 + 2\,(1 - x^3 - y^3)\, q^{\frac{3}{2}} + 6\,\sqrt{q} - 6\,q\ . \end{aligned} \qquad (y = z - \sqrt{q})$$

with initial values $z(0) = 1$, $q(0) = 1$.

The function $y' = p$ has a singularity at $x = r$ iff $q(r) = 1$, i.e. one has to integrate until $q(x) = 0$. Note that for $q(x) = 0$ we have $z' = 2$, $q' = -2$, i.e. the critical value $x = r$ is passed with finite velocity, whence the transformation $(y,p) \to (z,q)$ regularizes the problem. One expects $q(x) \approx 2\,(r - x)$ and $z(r) \approx y(r) - 2\,(r - x)$ for x close to r.

Suppose we have integrated until $q(x) \le \epsilon$. Then

$$\epsilon \ge q(x) = q(x) - q(r) \approx q'(r)\,(x - r) \approx -2\,(x - r)\ ,$$

i.e., $r - x \approx q(x)/2 \le \epsilon/2$. Furthermore,

$$y(r) - z(x) = z(r) - z(x) \approx z'(r)(r - x) \approx 2\,(r - x) \le \epsilon\ ,$$

i.e, hence $z(x)$ approximates $y(r)$ up to ϵ.

Result: if $q(x) < \epsilon$, then $[x, z(x)]$ approximate $[r, y(r)]$ up to $\epsilon/2$ and ϵ, respectiverly.

Unfortunately the square roots in the new differential system prevent standard adaptive numerical integrators to work properly in the neighbourhood of $q = 0$ (the usual adaptive mechanisms will fail). We implement an ad-hoc adaptive scheme based on the standard fourth order Runge-Kutta method that decreases the stepsize when approaching $q = 0$.

```
>> M:=Dom::Matrix(): DIGITS:=17: precision:=10^(-14):
```

The procedure **RKStep** is an implementation of the usual fourth order Runge-Kutta method with stepsize h for the system $z' = Z(x,z,q)$, $q' = Q(x,z,q)$.

```
>> RKStep:=proc(x,z,q,h)
   local k1,k2,k3,k4;
   begin
     k1:=M(2,1,[Z(x,z,q), Q(x,z,q) ]);
     k2:=M(2,1,[Z(x+h/2,z+h/2*k1[1],q+h/2*k1[2]),
                Q(x+h/2,z+h/2*k1[1],q+h/2*k1[2])]);
     k3:=M(2,1,[Z(x+h/2,z+h/2*k2[1],q+h/2*k2[2]),
                Q(x+h/2,z+h/2*k2[1],q+h/2*k2[2])]);
     k4:=M(2,1,[Z(x+h  ,z+h  *k3[1],q+h  *k3[2]),
                Q(x+h  ,z+h  *k3[1],q+h  *k3[2])]);
     1/6*(k1+2*k2+2*k3+k4);
```

```
    x+h,z+h*%[1],q+h*%[2]
  end_proc:
```

The following procedure calls **RKStep** with an initial choice h of the stepsize. The stepsize is decreased by factors of 1/2 until the estimate of the local error is smaller than the global value *precision*. We must not use the result of **RKStep** if the computation of a square root has lead us out of the real numbers.

```
>> adaptiveRKStep:=proc(x,z,q,h)
   local result1,result2,z1,z2,q1,q2,err;
   # compare a single RKStep(h) against two steps with stepsize h/2. #
   # The usual adaptive choice of the new stepsize                    #
   #        h:=float( h* (precision/errorestimate)^(1/5) ):           #
   # for a fourth order scheme will not work in this  case for small #
   # values of q due to the square roots. So just try again with      #
   # smaller stepsize.                                                #
     begin
      result1:=RKStep(x,z,q,h):
      result2:=RKStep(RKStep(x,z,q,h/2),h/2):
      z1:=result1[2]: z2:=result2[2]:
      q1:=result1[3]: q2:=result2[3]:
      err:= sqrt( abs(z1-z2)^2 +abs(q1-q2)^2 ) ;
      if is(q1,Type::RealNum) and is(q2,Type::RealNum) and
         q1>0 and q2>0 and err<precision
         then (x+h,z2,q2,h)
         else adaptiveRKStep(x,z,q,h/2)
      end_if
     end_proc:


>> y:=func(z-sqrt(q),z,q);
   Z:=func(float( 2+  (1-x^3-y(z,q)^3)*q-3*sqrt(q)),x,z,q):
   Q:=func(float(-2+2*(1-x^3-y(z,q)^3)*q*sqrt(q)+6*sqrt(q)-6*q),x,z,q):

   # The initial data : #
   x0:=0:z0:=1.0:q0:=1.0:h0:=1.0: data:=x0,z0,q0,h0:

   # The Runge-Kutta iteration: #
   while data[3]>precision do
     data:=adaptiveRKStep(data)
   end_while:
   print(Unquoted, "Solution: r= ".expr2text(data[1]).
   ", y(r)= ".expr2text(data[2]));
   print(Unquoted, "Approximate error: ".expr2text(data[3]));
   print(Unquoted, "Minimal step size used: ".expr2text(data[4]));

           Solution: r= 1.64437669033878810, y(r)= 0.931938765110228140
```

```
Approximate error: 4.45198434211447520e-15
```

```
Minimal step size used: 5.68434188608080140e-14
```

References

[1] *M. Abramowitz, I.A. Stegun*, Handbook of Mathematical Functions, Dover, New York, 1982

[2] *R.T. Gregory, D.L. Karney*, A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, New York, 1969

[3] *J.R. Westlake*, A Handbook of Numerical Matrix Inversion and Solution of Linear Systems, John Wiley and Sons, New York, 1968

printed with
jdm
Version 0.63